Plataforma de experimentación de controladores
Manual de Usuario

Rafael Romero Vicente

25 de junio de 2001

Índice General

1 Ensayo de controladores **7**
1.1 Introducción 7
1.2 Creación del modelo en Simulink 8
1.3 Creación del modelo en Simulink a partir de un modelo base 12
1.4 Ensayo del controlador en el robot RM10 14
1.5 Cockpit y Trace 14
1.5.1 Introducción 14
1.5.2 Cockpit 15
1.5.3 Trace 18

2 Diario de Incidencias **21**
2.1 Introducción 21
2.2 Nivel software 21
2.2.1 dSPACE 21
2.2.2 Matlab y Simulink 21
2.2.3 Aplicaciones Específicas 24
2.3 Nivel hardware 24
2.3.1 Conexiones/cableado 24
2.3.2 Electrónica original 25
2.3.3 Tarjetas electrónicas 25

Índice de Figuras

1.1 Modelo de SIMULINK para prueba de controladores. 10
1.2 Esquema de trabajo especificando la programación de controladores en el RM10 a partir de un nuevo modelo. 11
1.3 Esquema de trabajo especificando la programación de controladores en el RM10 a partir de un modelo base. 13
1.4 Programa SCADA para la supervisión de ensayos con el Robot Manipulador de experimentación. 17
1.5 Aplicación para visualizar la evolución de variables definidas en el modelo de SIMULINK. 19

2.1 Enlaces del brazo mecánico del robot industrial RM-10 e indicación de sus movimientos posibles y el criterio de signos adoptado. 22
2.2 Vectores de orientación en la herramienta del manipulador con indicación de los ángulos roll, pitch y yaw y el convenio de sentidos positivos tomado. . . . 23

Capítulo 1

Instrucciones para el ensayo de Controladores

1.1 Introducción

Tal y como describe el capítulo 4 de la memoria del proyecto se ha generado una biblioteca de bloques para SIMULINK que facilita la construcción de entornos de simulación y experimentación seguros.

Los cuatro bloques necesarios para la generación de modelos de SIMULINK que ya han sido anteriormente descritos son resumidos a continuación:

- Bloque *Protecciones y Robot*, que vigilará la aparición de alarmas antes de permitir cualquier actuación.
- Bloque *Control*, dedicado a la implementación de los controladores desarrollados. El bloque *Control* está habilitado por el bloque *Protecciones y Robot* que se encarga de comprobar el correcto funcionamiento del sistema.
- Bloque *Generador de Trayectorias*, que se encargará de suministrar los vectores de referencia de posición y velocidad al bloque *Control*.
- Bloque *Conmutación del Control*, cuyo principal cometido será conmutar entre las distintas posibles señales de control en función del estado del sistema, demultiplexar el vector resultante y enviarlo a la tarjeta *dSPACE*.
- Bloque *Modelo RM-10*, que permite simular, de forma aproximada, el comportamiento del robot real para el ensayo de controladores.

En el presente capítulo se establecen los pasos a seguir para probar controladores en el robot RM10.

El proceso de prueba de controladores en SIMULINK consistirá en primer lugar en la elaboración de un modelo de SIMULINK para programar el controlador. Para ello se puede generar un modelo nuevo (como se detalla en la sección 1.2) o partir de un modelo base y modificarlo (de acuerdo con el proceso descrito en la sección 1.3). Una vez construido el modelo en SIMULINK, será compilado y cargado en la tarjeta DSP (de la compañía dSPACE)

instalada en el ordenador para la aplicación de las leyes de control sobre el RM-10. Finalmente, la evolución de los ensayos podrá ser monitorizada mediante las aplicaciones *Cockpit*, *Trace* o *Controldesk*.

1.2 Creación del modelo en Simulink

La programación se llevará a cabo mediante diagramas de bloques de SIMULINK. Básicamente, el procedimiento a seguir consistirá en copiar de la librería *RM10* los bloques *Protecciones y Robot*, *Generador de Trayectorias* y *Conmutación del Control* e implementar el bloque *Control* que incluirá el controlador desarrollado por el usuario. Seguidamente, habrá que conectar los bloques entre sí adecuadamente y compilar el programa obteniendo el código objeto que se ejecutará en la tarjeta dSPACE.

A continuación se detallan los pasos a seguir para elaborar un modelo de controlador para el RM10:

Paso 1 Ejecutar el programa MATLAB desde el menú de inicio. Una vez en él, seleccionaremos nuestro directorio de trabajo: File→Set Path→Browse. Por ejemplo el directorio: $c:\backslash usuario\backslash RM10\backslash Matlab\backslash Simulink\backslash rm10par$. Este paso ha sido facilitado con el script *cdi.m*. Ejecutando este archivo aparece un cuadro de dialogo con el clásico árbol de directorios de Windows que permite cambiar fácilmente de directorio mediante la elección de cualquier fichero alojado en él. Análogamente, el comando *ini.m* cambia automáticamente al directorio *rm10par* citado anteriormente (destinado al ensayo de controladores en el robot real).

Paso 2 Iniciar SIMULINK. Aparecerá la ventana *Library:simulink* donde, además de las tradicionales librerías, aparece la librería RM10 (identificada por una imagen del robot) que incluye los bloques *Protecciones y Robot*, *Generador de Trayectorias* y *Conmutación del Control* descritos anteriormente.

Paso 3 Crear un nuevo modelo de SIMULINK: File→New→Model. Generará una ventana *untitled*. En este punto resulta conveniente dar un nombre al modelo. Para permitir el correcto funcionamiento de la aplicación *Posición Absoluta* que, residente en memoria, garantiza mantener la posición real del robot, es necesario nombrar el modelo como *RM10par.mdl* en el citado directorio:
$c:\backslash usuario\backslash RM10\backslash Matlab\backslash Simulink\backslash rm10par$

Paso 4 Incluir los bloques *Protecciones y Robot*, *Generador de Trayectorias* y *Conmutación del Control* en el modelo. Para ello, abra la librería RM10 haciendo doble click sobre su icono correspondiente dentro de la ventana *Library:simulink*. Encontrará los tres bloques necesarios que hay que incluir en la ventana del modelo.

Paso 5 Elaborar el bloque *Control*. Se creará un subsistema al que llamaremos Control. Con este fin copiaremos en nuestro modelo un bloque *Out1* de la librería de SIMULINK *Connections*. Seleccionaremos con una ventana el bloque *Out1* copiado y a continuación Edit→Create Subsystem. Le cambiaremos el nombre de bloque *Subsystem* por *Control* y borraremos el bloque exterior *Out1*. Finalmente abriremos el bloque *Control* creado

y copiaremos en él un bloque *Out1* (que renombraremos como U_Controlador) un bloque Enable y los bloques *In1* necesarios para nuestro control. Estos bloques pueden encontrarse en la librería de SIMULINK *Connections*.

En el ejemplo mostrado en la figura 1.1 se incluyeron 4 bloques *In1* de entrada renombrándolos así: In1→q_ref, In2→qd_ref, In3→q, In4→qd. Y un bloque de salida: Out1→U_Controlador. Las señales que podemos incluir en el bloque de control son descritas a continuación:

q_ref/r_ref vector de referencia de posición articular/lineal procedente del bloque *Generador de Trayectorias*.

qd_ref/rd_ref vector de referencia de velocidad articular/lineal procedente del bloque *Generador de Trayectorias*.

q vector de posición articular procedente del bloque *Protecciones y Robot*. Suministra la posición de las articulaciones del RM10 proporcionadas por los encoder.

qd vector de velocidad articular procedente del bloque *Protecciones y Robot*. Proporciona la velocidad de las articulaciones del RM10 generadas derivando las lecturas de posición de los encoder.

r vector de posición lineal procedente del bloque *Protecciones y Robot*. Suministra la posición cartesiana y la orientación de la garra del RM10.

rd vector de velocidad lineal procedente del bloque *Protecciones y Robot*. Proporciona la velocidad de la garra del robot respecto al sistema inercial de la base del robot.

Enable señal para habilitar el control por parte del bloque *Protecciones y Robot*. Irá conectada a CONTROLADOR_ACTIVO del bloque *Protecciones y Robot*.

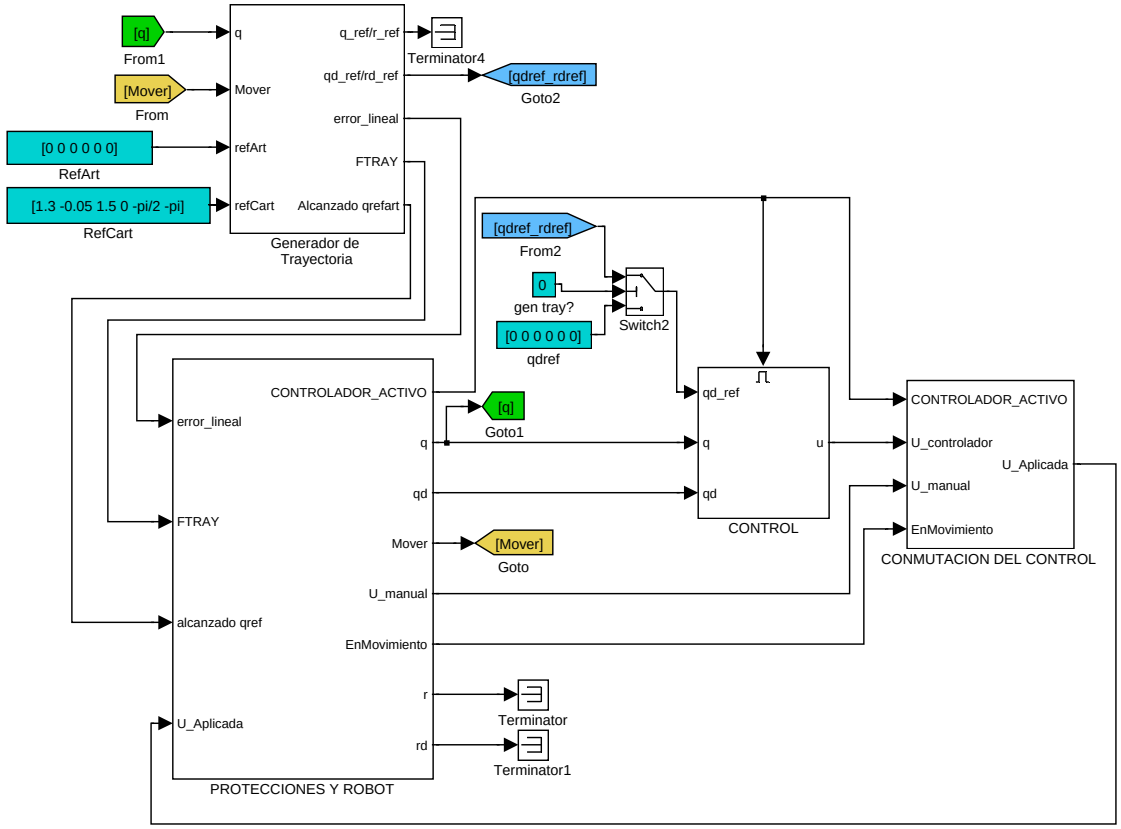
U_controlador vector de señales de control que conectaremos al bloque *Conmutación del Control*.

Paso 6 Conectar los cuatro bloques entre sí, tal y como se detalla en la tabla 1.1 y como puede verse en la figura 1.1. En dicha figura, además, se ha incluido la posibilidad de fijar una referencia nula a través del Switch2 aunque esto no es siempre necesario y podría ser prescindible según el caso. Las señales no utilizadas en nuestro bloque *Control* pueden conectarse a bloques *Terminator* de la librería *Connections* como ocurre con *r* y *rd* en el ejemplo. Finalmente se incluirán en el modelo los vectores de referencia articular y lineal que se conectarán a las entradas correspondientes del bloque *Generador de Trayectorias*. Para ello se ha de copiar dos veces el bloque *Constant* de la librería *Sources*. Como se observa en la figura 1.1 el uso de etiquetas puede facilitar la tarea de establecer conexiones sin excesivas líneas que crearían confusión.

Paso 7 Compilar el modelo. Finalizada la elaboración del controlador procederemos a compilar el modelo con el nuevo controlador, volcándolo en la tarjeta *dSPACE*. Menú Tools→RTWBuild.

Señal	Origen	Destino
q_ref/r_ref	*Generador de Trayectorias*	*Control*
qd_ref/rd_ref	*Generador de Trayectorias*	*Control*
error_lineal	*Generador de Trayectorias*	*Protecciones y Robot*
FTRAY	*Generador de Trayectorias*	*Protecciones y Robot*
Alc. qrefart	*Generador de Trayectorias*	*Protecciones y Robot*
CONTROLADOR_ACTIVO	*Protecciones y Robot*	*Control*(Enable)
CONTROLADOR_ACTIVO	*Protecciones y Robot*	*Conmutación del Control*
q	*Protecciones y Robot*	*Control*
r	*Protecciones y Robot*	*Control*
q	*Protecciones y Robot*	*Generador de Trayectorias*
qd	*Protecciones y Robot*	*Control*
rd	*Protecciones y Robot*	*Control*
Mover	*Protecciones y Robot*	*Generador de Trayectorias*
U_manual	*Protecciones y Robot*	*Conmutación del Control*
EnMovimiento	*Protecciones y Robot*	*Conmutación del Control*
U_Aplicada	*Conmutación del Control*	*Protecciones y Robot*

Tabla 1.1: Conexiones entre los bloques *Protecciones y Robot*, *Generador de Trayectorias*, *Conmutación del Control* y *Control*.



:

Figura 1.1: Modelo de Simulink para prueba de controladores.

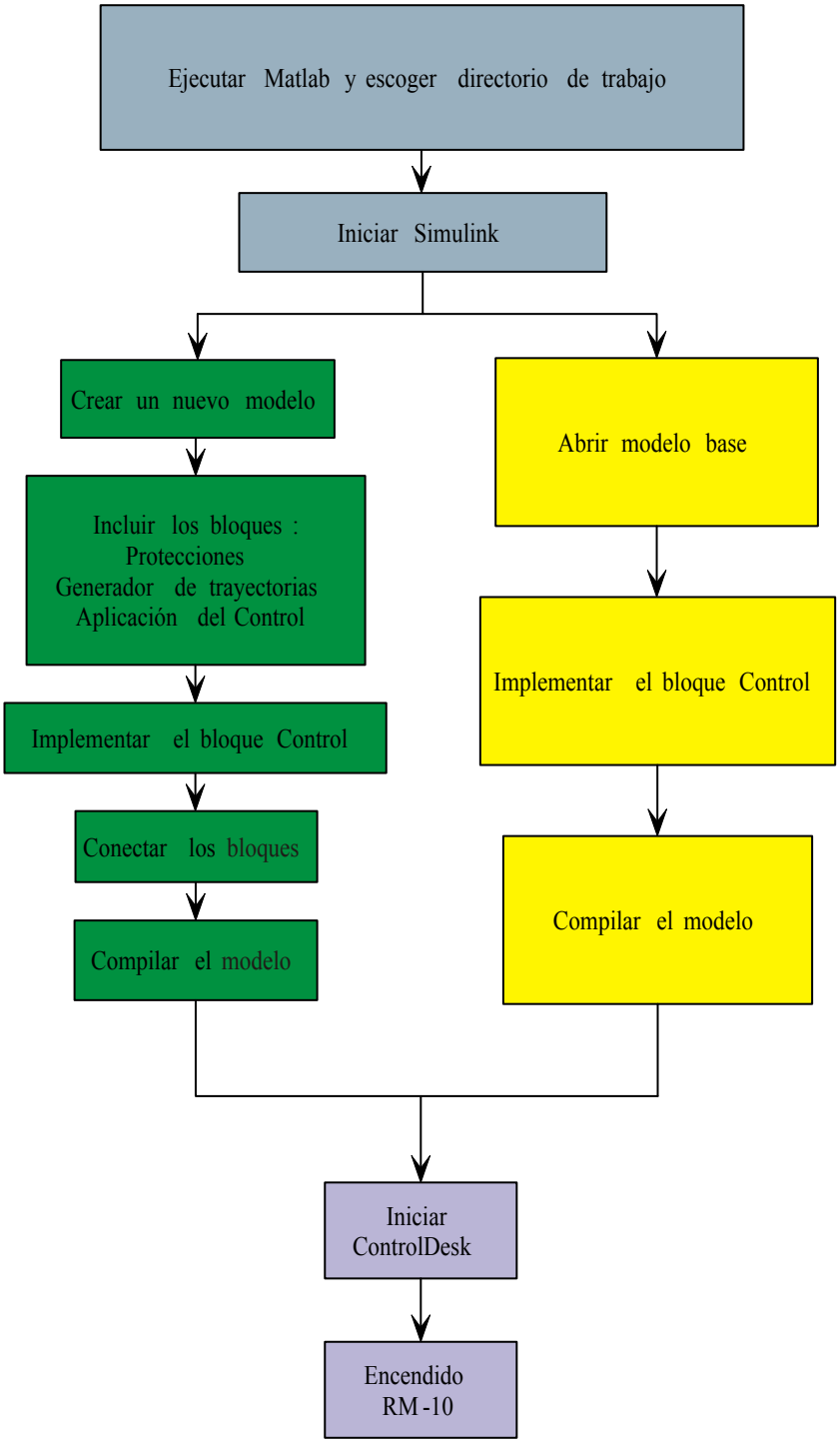


Figura 1.2: Esquema de trabajo especificando la programación de controladores en el RM10 a partir de un nuevo modelo.

1.3 Creación del modelo en Simulink a partir de un modelo base

El procedimiento a seguir consistirá en modificar el bloque de control en un modelo de SIMULINK ya existente y que servirá de base para la programación, llamado RM10. Una vez implementado el nuevo controlador se compilará el programa.

A continuación se detallan los pasos a seguir:

Paso 1 Ejecutar el programa MATLAB desde el menu de inicio. Una vez en él seleccionaremos nuestro directorio de trabajo: File→Set Path→Browse. Por ejemplo el directorio *c:\usuario\RM10\Matlab\Simulink\rm10par*.

Paso 2 Iniciar SIMULINK. Aparecerá la ventana *Library:simulink* donde, además de las tradicionales librerías, aparece la librería RM10 que incluye los bloques *Protecciones y Robot*, *Generador de Trayectorias* y *Conmutación del Control* descritos anteriormente.

Paso 3 Abrir el modelo RM10. File→Open y seleccionar el archivo RM10 dentro del directorio *c:\Usuario\Rm10*. Se generará una ventana *RM10* con el modelo base. En este punto resulta conveniente dar un nombre a nuestro modelo: Menú File→Save As y escoger un nombre adecuado. Para permitir el correcto funcionamiento de la aplicación *Posición Absoluta* que, residente en memoria, garantiza mantener la posición real del robot, es necesario nombrar el modelo como *RM10par.mdl*.

Paso 4 Implementar el bloque *Control*. Abra el bloque *Control* haciendo doble click sobre él. Encontrará que se encuentra prácticamente vacío conteniendo únicamente (además de un bloque Enable para su activación por parte del bloque *Protecciones y Robot*) las entradas y la salida que se detalla a continuación:

q_ref vector de referencia de posición articular/lineal procedente del bloque *Generador de Trayectorias*.

qd_ref vector de referencia de velocidad articular/lineal procedente del bloque *Generador de Trayectorias*.

q vector de posición articular procedente del bloque *Protecciones y Robot*. Proporciona la posición de las articulaciones del RM10 proporcionadas por los encoder.

qd vector de velocidad articular procedente del bloque *Protecciones y Robot*. Suministra la velocidad de las articulaciones del RM10 generadas derivando las lecturas de posición de los encoder.

U_controlador vector de señales de control que conecta con el bloque *Conmutación del Control*.

Una vez dentro del bloque *Control* implemente el controlador deseado a partir de las señales de entrada de dicho bloque y generando la señal de control a aplicar (U_controlador).

Paso 5 Compilar el modelo. Finalizada la elaboración del controlador procederemos a compilar el modelo con el nuevo controlador, volcándolo en la tarjeta *dSPACE*. Menú Tools→RTWBuild.

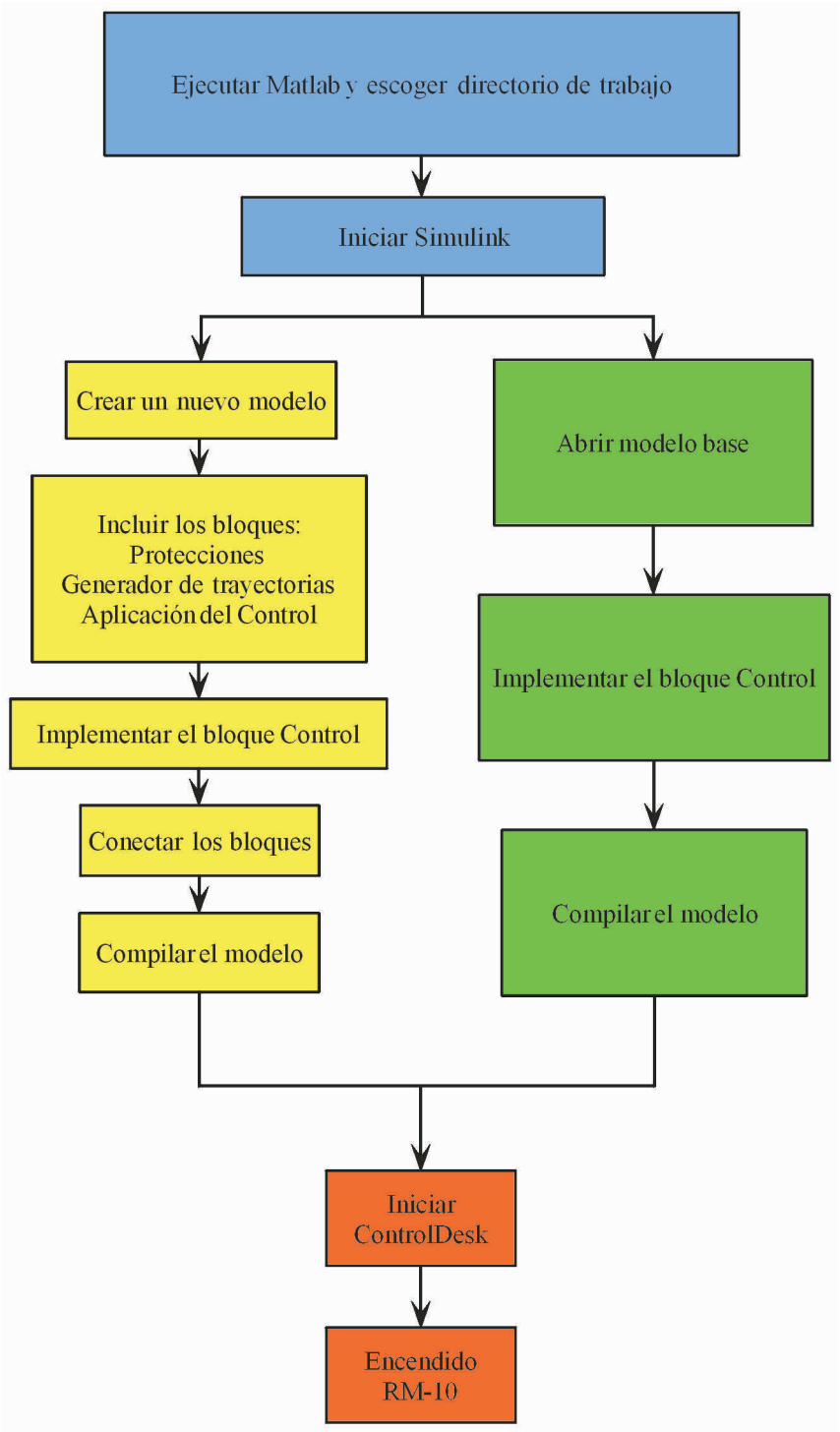


Figura 1.3: Esquema de trabajo especificando la programación de controladores en el RM10 a partir de un modelo base.

1.4 Ensayo del controlador en el robot RM10

Una vez elaborado el modelo de control y descargado en la tarjeta dSPACE de acuerdo con la sección anterior, la tarjeta comienza la ejecución del programa pudiendo cerrarse MATLAB. A continuación hay que ejecutar el programa *Cockpit* (el SCADA de la *dSPACE* que permitirá supervisar las pruebas del nuevo controlador sobre el robot) y finalmente se procederá al encendido del RM10 con la siguiente secuencia:

1 Girar la llave de potencia hasta la posición 1

2 Pulsar el botón verde *START*. Se podrá oir el sonido característico de carga de un condensador procedente del armario.

3 Girar la llave *control* hasta la posición *control PC*.

Adicionalmente es posible insertar nuevos controles de edición y monitorización en el programa *Cockpit* de forma que el usuario puede establecer su propio entorno para el ensayo en cuestión. No obstante se recomienda mantener la base del panel de ejemplo *rm10par.ccs* que contiene los controles principales para desarrollar experimentos seguros.

Para la obtención de representaciones gráficas en el dominio del tiempo de variables asociadas al controlador ensayado puede hacerse uso de la aplicación *Trace* de dSPACE.

Las aplicaciones *Cockpit* y *Trace* son comentadas en detalle en la sección posterior. Alternativamente puede utilizarse la aplicación *Controldesk* que engloba funcionalidades de ambos programas bajo un mismo entorno.

1.5 Cockpit y Trace

1.5.1 Introducción

Para establecer una interface gráfica que permita al usuario interactuar con el modelo de SIMULINK y supervisar los ensayos en el robot manipulador se hace necesaria la utilización de las aplicaciones *Cockpit* y *Trace*.

Cockpit es un programa SCADA que se proporciona con la adquisición de la tarjeta DSP y que permite, mediante un editor gráfico, diseñar pantallas de explotación de variables del modelo así como su modificación en tiempo real.

En la aplicación Trace de *dSPACE* se representa la evolución de cualquier variable definida previamente en el modelo de SIMULINK. Esta herramienta permite representar gráficamente valores relevantes del ensayo y comprobar el funcionamiento del controlador.

También es útil para el uso de la plataforma el programa Controldesk. Esta aplicación es una alternativa al uso de los programas anteriores para la monitorización y adquisición de datos, si bien algunas de las limitaciones que su utilización impone determinó optar por Cockpit y Trace. No obstante, esta aplicación dispone de una función extra que no poseen sus antecesoras y es la posibilidad de cargar un modelo ya compilado, directamente en la tarjeta sin necesidad de ser recompilado. Esta ventaja sí es explotada en el aprovechamiento de la plataforma.

1.5.2 Cockpit

Esta aplicación, distribuida por la firma *dSPACE* conjuntamente con sus tarjetas DSP, permite llevar a cabo las tareas de supervision durante los ensayos con el RM. Concretamente, el programa permite:

- visualizar medidas, como por ejemplo: lecturas de encoders, medidas de posición, error en posición y señal de control,estado del robot, alarmas, etc...
- Modificar variables del modelo de SIMULINK. A través de elementos de edición se pueden variar parámetros del controlador, ordenar movimientos, transicionar entre estados del robot, introducir valores numéricos correspondientes a referencias finales en posición (tanto en coordenadas articulares como lineales),...

Para diseñar pantallas de explotación específicas para la plataforma, se hizo uso del editor gráfico del programa. El proceso de generación es bastante sencillo y consiste básicamente en insertar los distintos controles necesarios que se agrupan bajo las categorías:

- Input Controls.
- Output Controls.
- Input/Output Controls.
- Special Controls.

La mayoría de los controles incluidos requieren la asignación de las variables que van a ser monitorizadas o bien sobre las que serán aplicadas las modificaciones introducidas. Adicionalmente es posible etiquetarlos para identificar sus funciones mediante textos suficientemente identificativos.

Se fabricaron distintos paneles de control en función del tipo de ensayos que se requerían. Estos paneles, una vez creados, pueden ser salvados como ficheros .ccs para su posterior reutilización. No todos los controles incluidos en los diferentes paneles son distintos, muy al contrario, comparten partes comunes ubicadas en la zona superior de la pantalla. Estas secciones pueden ser agrupadas en las siguientes categorías:

- **Generación de movimientos Manuales/Automáticos.** Con los pulsadores de posicionamiento manual pueden ordenarse movimientos basados en referencias en velocidad (tanto positivas como negativas) constante. La velocidad puede escogerse en forma de porcentaje de las máximas admisibles en un control del tipo *incremental input* alojado en el panel. Para la generación de movimientos automáticos articulares se disponen controles de este mismo tipo que permiten la introducción de posiciones de referencia articular finales. También se incorporan displays para visualizar las posición actual y el par aplicado en cada articulación.
- **Entradas/Salidas Digitales.** Se corresponden con indicadores luminosos (ver sección 3.4 en la memoria del proyecto). Las entradas monitorizadas son EM6, IFR, IN8, IN9, IN10 y *Verificación Control PC* y las salidas *Enable*, *Brake* y *CIR5*.

- **Máquina de Estados.** En este apartado se engloban los indicadores luminosos representativos de los distintos estados del robot. Asimismo se añaden pulsadores para habilitar al RM(paso de encendido a operación), para conmutar los modos de movimiento (manual/automático) y los modos de trayectorias (articular/cartesiano), para iniciar un movimiento automático o para posicionar el robot en su posición *Home*.

- **Coordenadas Cartesianas.** Permite visualizar la posición cartesiana y la orientación de la herramienta actuales (respecto del sistema de referencia inercial de la base del robot). Además es posible introducir posiciones de referencia finales para la generación de movimientos en el espacio cartesiano.

- **Supervisión de Alarmas.** Detecta la aparición de alguna de las alarmas programadas:

  - Error excesivo en trayectoria lineal.
  - Límite de espacio de trabajo alcanzado.
  - Límite de velocidad alcanzado.
  - Límite ángulo de articulaciones alcanzado.
  - Sin control desde PC.
  - EM6 desactivado.
  - Fallo en los InX.
  - Frenos activados.
  - Sin movimiento.
  - Mal Frenado.

  Estos indicadores de alarmas, una vez desaparecida la causa que las provocó, permanecerán encendidas (para facilitar su supervisión) hasta que no se pase al estado de operación (pulsando *Habilitar*). Además algunas de ellas pueden deshabilitarse momentáneamente para permitir, por ejemplo, reintroducir al brazo en el volumen de trabajo cartesiano o articular.

- **Finalizar aplicación.** Este pulsador es el encargado de comunicar a la aplicación *Posición Absoluta* (residente en memoria) que hemos finalizado los ensayos. Provoca el reseteo de la tarjeta por parte de esta aplicación lo que exigirá previamente la finalización de la animación de Cockpit.

Adicionalmente, pueden encontrarse en los paneles SCADA desarrollados otro tipo de controles como: visualización de pulsos de encoder, pulsador de reseteo de los mismos, cajas de texto para la sintonización de parámetros de controladores PID, secciones para la generación de referencias de posición y velocidad (pudiendo elegir entre diferentes formas de ondas), monitorizadores de modelos de Lugre experimentados,...

En la figura 1.4 aparece una composición tipo de panel de control de Cockpit.

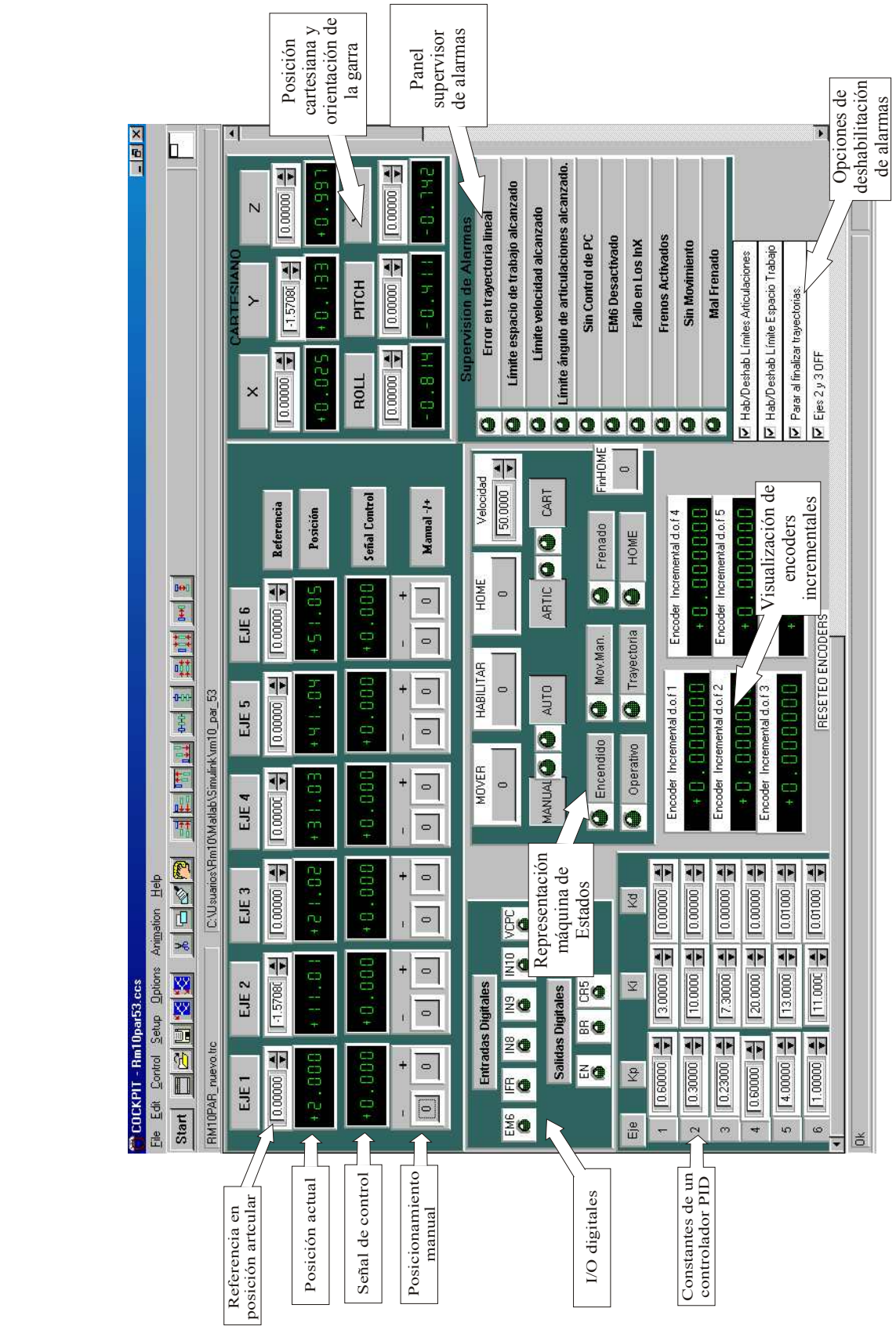


Figura 1.4: Programa SCADA para la supervisión de ensayos con el Robot Manipulador de experimentación.

1.5.3 Trace

La aplicación anterior carece de la facultad de representar gráficamente señales capturadas durante los ensayos. Este es, precisamente, el cometido de la aplicación Trace.

Trace hace uso del archivo .TRC que genera toda compilación de controladores en la tarjeta. Mediante este fichero obtiene un mapa de las variables implicadas en el programa que se está ejecutando en la tarjeta, agrupadas en los distintos bloques de las que forman parte.

La ventana principal del programa permite la elección de las variables principales que se pretenden representar. Cada elección genera una nueva ventana en la plantilla (Template) que se encuentre habilitada en ese momento. Una plantilla es un entorno que permite agrupar distintas ventanas para la representación de variables. Además, en la ventana principal se puede elegir el tiempo de duración de la captura, comenzar la captura de forma manual mediante la activación del botón *Start* o hacerlo automáticamente escogiendo una señal que haga de Trigger para establecer el inicio de captura. Asimismo es posible exigir al programa la reactivación de la captura tras su finalización mediante la habilitación de la opción *Auto*. Comúnmente, se ha hecho uso de la señal *En Movimiento* para su utilización como trigger.

El usuario puede generar tantas plantillas como desee y en cada ventana de una plantilla pueden solaparse varias representaciones gráficas de las variables deseadas. En el eje de abscisas, en las capturas obtenidas, usualmente se representa el tiempo (medido en segundos a partir del inicio de la captura) aunque es posible generar gráficas del tipo XY en las que la abscisa represente cualquier otra variable. Se ha hecho uso de este tipo de gráfica para la representación del par de control (señal u) frente a la velocidad (qd) articular para la identificación de modelos de fricción.

En la figura 1.5 puede observarse la pantalla donde se representan los valores de posición angular, velocidad angular, velocidad de referencia y señal de control para los ejes 1 a 6. En concreto, dicha figura representa la evolución de las citadas variables ante un movimiento conjunto de todas las articulaciones al mismo tiempo (movimiento *Home*).

Una vez finalizada una captura, ésta puede ser almacenada en ficheros .mat para su posterior utilización desde la aplicación matemática Matlab. Las variables se guardan en dichos archivos de acuerdo a la siguiente estructura de datos:

trace_x_n. Contiene el nombre de la variable almacenada en el eje de abscisas (en la mayoría de los casos el tiempo).

trace_y_n. Establece la lista de nombres de las variables almacenadas como eje de ordenadas. El criterio general establecido para la elección de variables más oportunas produce el siguiente conjunto de variables:

- 1.model Root/CONMUTACION DEL CONTROL/B:Saturation1
- 2.model Root/CONMUTACION DEL CONTROL/B:Saturation2
- 3.model Root/CONMUTACION DEL CONTROL/B:Saturation3
- 4.model Root/CONMUTACION DEL CONTROL/B:Saturation4
- 5.model Root/CONMUTACION DEL CONTROL/B:Saturation5
- 6.model Root/CONMUTACION DEL CONTROL/B:Saturation6

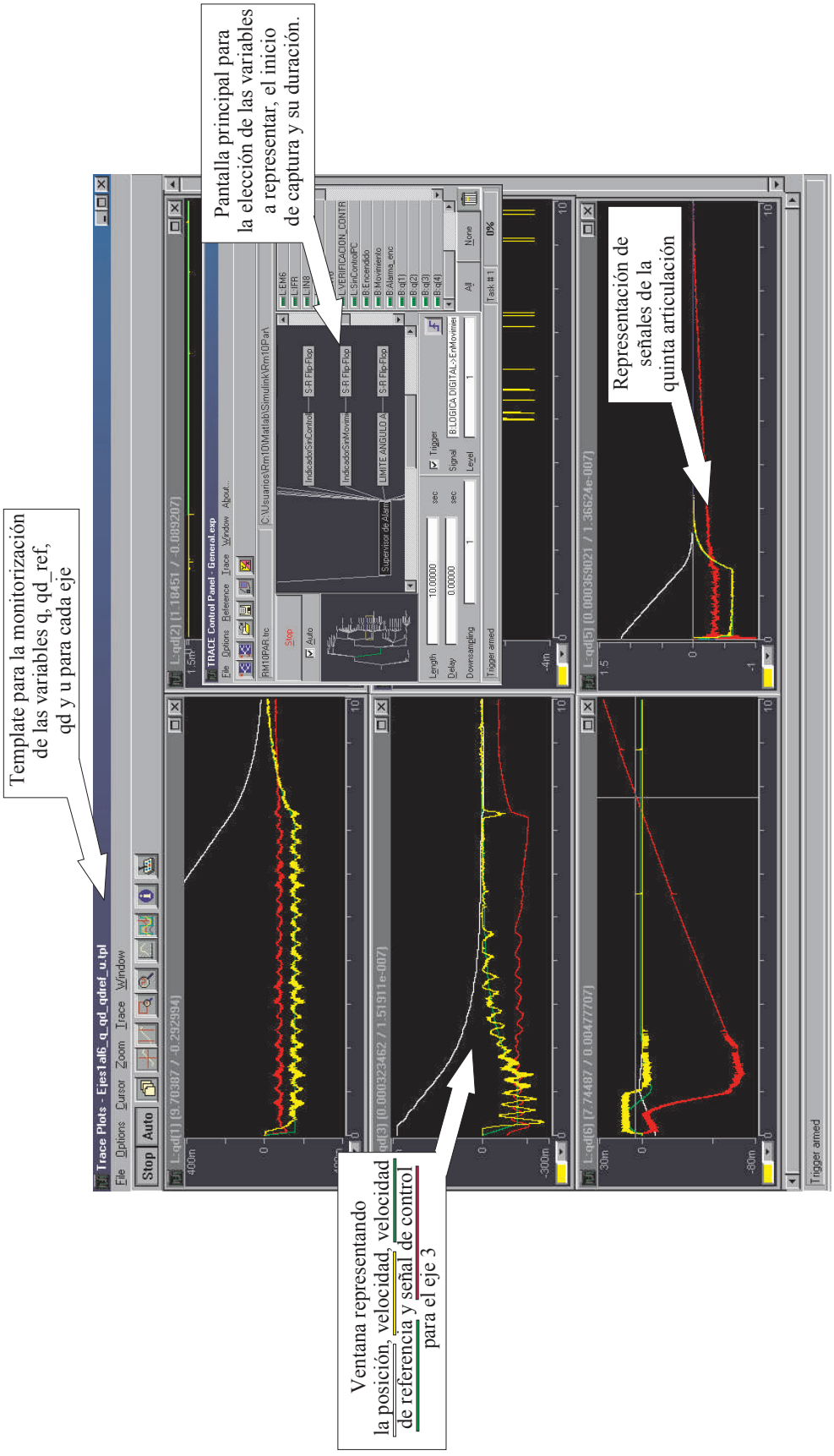


Figura 1.5: Aplicación para visualizar la evolución de variables definidas en el modelo de SIMULINK.

- 7.model Root/PROTECCIONES Y ROBOT/L:q(1)
- 8.model Root/PROTECCIONES Y ROBOT/L:q(2)
- 9.model Root/PROTECCIONES Y ROBOT/L:q(3)
- 10.model Root/PROTECCIONES Y ROBOT/L:q(4)
- 11.model Root/PROTECCIONES Y ROBOT/L:q(5)
- 12.model Root/PROTECCIONES Y ROBOT/L:q(6)
- 13.model Root/PROTECCIONES Y ROBOT/L:qd(1)
- 14.model Root/PROTECCIONES Y ROBOT/L:qd(2)
- 15.model Root/PROTECCIONES Y ROBOT/L:qd(3)
- 16.model Root/PROTECCIONES Y ROBOT/L:qd(4)
- 17.model Root/PROTECCIONES Y ROBOT/L:qd(5)
- 18.model Root/PROTECCIONES Y ROBOT/L:qd(6)
- 19.model Root/PROTECCIONES Y ROBOT/Manual/L:qd_ref(1)
- 20.model Root/PROTECCIONES Y ROBOT/Manual/L:qd_ref(2)
- 21.model Root/PROTECCIONES Y ROBOT/Manual/L:qd_ref(3)
- 22.model Root/PROTECCIONES Y ROBOT/Manual/L:qd_ref(4)
- 23.model Root/PROTECCIONES Y ROBOT/Manual/L:qd_ref(5)
- 24.model Root/PROTECCIONES Y ROBOT/Manual/L:qd_ref(6)

trace_x. Proporciona el vector de valores de la variable de abscisas (usualmente, el tiempo en segundos)

trace_y. Constituye el grupo fundamental, una matriz de valores de los datos donde el primer índice diferencia las distintas variables y el segundo las magnitudes para cada instante de tiempo. Por ejemplo, trace_y(3,2000) representa el valor de la señal de par sobre el eje 3 a los 2 segundos de comenzar el ensayo.

Capítulo 2

Diario de Incidencias

2.1 Introducción

En el presente capítulo se documenta todas aquellas incidencias acaecidas durante la elaboración del proyecto con la finalidad de servir de guía para futuros desarrollos sobre la plataforma de experimentación de controladores.

2.2 Nivel software

Dentro del nivel software se incluye tanto las aplicaciones dSPACE (*Cockpit*, *Trace* y *Controldesk*) como Matlab y SIMULINK y las aplicaciones desarrolladas específicamente: *Posición Absoluta* y *3D RM-10*.

2.2.1 dSPACE

Es necesario configurar Matlab para compilar modelos en la dSPACE en función del software de monitorización que va a utilizarse: *Cockpit/Trace* o *Controldesk*. Concretamente, desde el prompt de Matlab, debe introducirse alguna de las siguientes instrucciones de configuración tras su instalación:

- >> rti_option TRCFileSyntax TRACE (para la utilización de *Cockpit/Trace*)
- >> rti_option TRCFileSyntax ControlDesk (para la utilización de *Controldesk*)

Por otro lado, se detecta un problema durante el uso de las aplicaciones *Cockpit* y *Controldesk* en el cálculo del Cos(0.0). Por parte de la tarjeta DSP se obtiene un valor que fluctúa, tomando valores alternativamente de '1' y '2' durante su monitorización con estas aplicaciones. Sin embargo, parece tratarse simplemente de un error de visualización y no un error de cálculo de la propia tarjeta por lo que no reviste excesiva gravedad.

2.2.2 Matlab y Simulink

Se produce un conflicto al intentar compilar un modelo en SIMULINK si se encuentra activa la protección contra virus *Norton Antivirus 2001*. Si no se desactiva antes *Norton Antivirus*

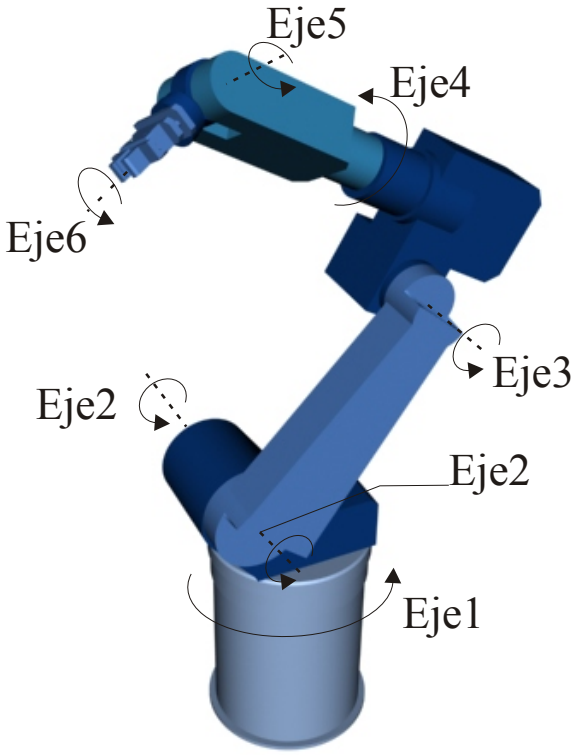


Figura 2.1: Enlaces del brazo mecánico del robot industrial RM-10 e indicación de sus movimientos posibles y el criterio de signos adoptado.

se puede provocar una situación de inestabilidad en el sistema que incluso puede causar el bloqueo total del ordenador y hacer inevitable el reseteo. La solución más sencilla es utilizar algún otro software antivirus o seguir utilizando Norton Antivirus 2001 pero desactivarlo temporalmente durante la compilación. No se descartan posibles futuras incompatibilidades con otros programas.

Una fuente de posibles errores consiste en la adopción de dos criterios distintos para el origen de medición de ángulos en el movimiento del eje 2. En el proceso de ajuste de cero se recurre a la alineación de las marcas adosadas al brazo manipulador destinadas a tal efecto por lo que parece conveniente tomar esta situación como origen de ángulos. Este criterio, que llamaremos *criterio RM10* ,ha sido el considerado en todo el esquema de protecciones y por consiguiente es el que debe emplear el usuario que desee desarrollar nuevos controladores. Por el contrario, en algunas de las funciones S heredadas del proyecto de Carlos Pérez precursor del presente, se toma la situación de marcas alineadas (eje 2 perpendicular al plano horizontal) como una posición de -$\pi/2$ radianes para el eje 2. No obstante, es posible hacer uso de las antiguas funciones así como de los resultados del modelo analítico (ver capítulo 2 de la memoria) que dependan de la posición con la sencilla precaución de establecer el siguiente cambio de variable:

$$\theta_2(Carlos) = \theta_2(RM10) + 3 * pi/2$$

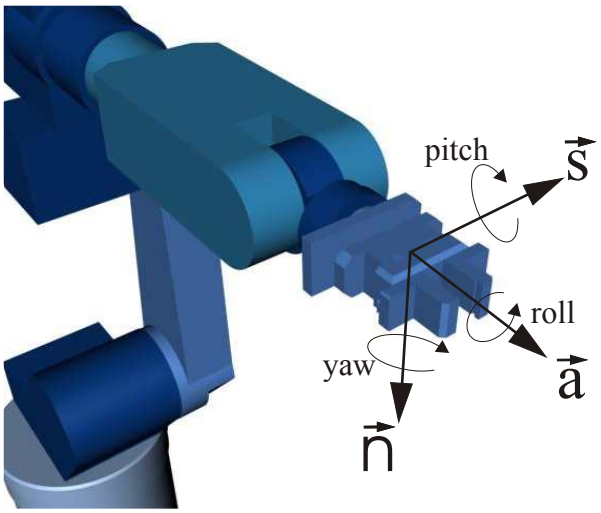


Figura 2.2: Vectores de orientación en la herramienta del manipulador con indicación de los ángulos roll, pitch y yaw y el convenio de sentidos positivos tomado.

Otro criterio importante seguido en el proyecto es la adopción de un criterio de signos adecuado para la representación de los pares de fuerza, posiciones, velocidades y aceleraciones. Éste se detalla en las figuras 2.1 y 2.2. Resumidamente se toma como referencia la marcada por las regletas adosadas al RM10 para el ajuste de cero. De esta forma, los sentidos de posiciones y pares positivos coinciden con el sentido de giro positivo del eje del motor. Las velocidades y aceleraciones tendrán pues su signo impuesto por las variaciones de posición. Desafortunadamente, el criterio adoptado en la electrónica original del RM10 es distinto. Concretamente, en cuanto al signo de los pares en los ejes 3 y 6, es el contrario al comentado. Análogamente, para las posiciones de los ejes 1, 2, 4 y 5, en el armario se considera positivo el sentido contrario al considerado anteriormente. Todo esto hace necesaria la inclusión de un cambio de signo en las correspondientes medidas de posiciones y aplicaciones de pares:

$$\begin{aligned}
\theta_1(Simulink) &= -\theta_1(RM10) \\
\theta_2(Simulink) &= -\theta_2(RM10) \\
\theta_3(Simulink) &= \theta_3(RM10) \\
\theta_4(Simulink) &= -\theta_4(RM10) \\
\theta_5(Simulink) &= -\theta_5(RM10) \\
\theta_6(Simulink) &= \theta_6(RM10)
\end{aligned} \tag{2.1}$$

$$\begin{aligned} u_1(Simulink) &= u_1(RM10) \\ u_2(Simulink) &= u_2(RM10) \\ u_3(Simulink) &= -u_3(RM10) \\ u_4(Simulink) &= u_4(RM10) \\ u_5(Simulink) &= u_5(RM10) \\ u_6(Simulink) &= -u_6(RM10) \end{aligned} \tag{2.2}$$

Aunque este problema ha sido solucionado en el bloque *Encoders* y en el bloque *Conmutación del Control* para hacerlo transparente al usuario final, fue una fuente de conflictos en las primeras etapas de desarrollo y deberá ser considerado en posibles modificaciones futuras de los bloques implicados.

2.2.3 Aplicaciones Específicas

La aplicación *Posición Absoluta* puede provocar el incorrecto funcionamiento de la aplicación *Cockpit*. Ocurrió en alguna ocasión que durante la ejecución de *Cockpit* este comunicó un error a la hora de intentar bloquear la aplicación que se estaba ejecutando en la dSPACE. Este error, de todas formas, es muy poco usual y se resuelve fácilmente reiniciando *Cockpit*.

2.3 Nivel hardware

Las posibles situaciones causantes de error y debidas a problemas hardware se comentan en esta sección. Las fuentes de error posibles se han estructurado en los siguientes campos:

- Conexiones/cableado.
- Electrónica original en el armario del RM-10.
- Tarjetas electrónicas desarrolladas específicamente.

2.3.1 Conexiones/cableado

La desconexión accidental del cable resolver o un mal contacto en la tarjeta resolver/encoder puede provocar el error en la lectura de posición, fácilmente detectable por dispararse el número de pulsos de encoder medidos por la dSPACE lo que genera una alarma programada en el bloque de *Protecciones y Robot* a tal efecto.

En la etapa Hardware de interfase, la electrónica correspondiente al armario del RM10 se encuentra desacoplada de la del PC. Esta última está alimentada directamente de la propia fuente de alimentación del PC anfitrión de la tarjeta dSPACE. La accidental desconexión del cable de alimentación procedente del ordenador puede causar la inoperabilidad de todo el sistema.

2.3.2 Electrónica original

A pesar de que los pasos para la habilitación de los servos son correctos y parece llegar señal de control a los mismos, el robot no se mueve. Una posible causa es que los frenos no estén siendo liberados correctamente (clemas de habilitación de frenos individuales abiertas, la señal de liberación de frenos no es la adecuada...). Otra situación que aconteció en el transcurso del proyecto y que causó este comportamiento fue el fundido de uno de los fusibles del armario del RM10 (concretamente el F5 en la parte anterior, junto a los contactores CP y CF) lo que se solucionó con la compra y sustitución del mismo. El modelo de fusible que requiere el RM10 es el UNE 21.103 de dimensiones 10x38, 1A. gl-gC, 500 V y I1=120 kA y existen repuestos disponibles en el armario destinado al material y documentación relacionados con el proyecto.

Para poder realizar ensayos con el controlador original del RM10 y hacer uso de la pistola de programación es necesario trabajar con los servos en modo velocidad mientras que para controlar mediante el PC se requiere el modo par. Para conocer en detalle ambos modos de funcionamiento y la forma de conmutar entre ellos puede consultarse el capítulo 1 de la memoria.

Con la pulsación del botón de STOP del armario del RM10 se produce un error en la lectura de los encoders que se disparan a un valor elevado y aleatorio que falsearía la posición real del robot. Para subsanarlo se modificó la aplicación *Posición Absoluta* y el bloque *Protecciones y Robot* del modelo de SIMULINK para capturar en el fichero *RM10PAR_buffer.dat* la posición del robot tras cada movimiento y recuperarla al habilitar el RM10 (en el paso al estado de Operación).

2.3.3 Tarjetas electrónicas

La anterior situación también puede ser provocada por un incorrecto contacto en la tarjeta resolver/encoder debido a un posible deterioro de alguna pista. Esto ha sido solucionado con la sustitución de dos de las 3 tarjetas antiguas (las más problemáticas) y el puenteado de alguna pista conflictiva en la tercera.

Si la llave de conmutación del control, que permite el control desde el PC, no se encuentra completamente girada, puede no provocarse la conmutación del relé correspondiente e impedir la habilitación del sistema (la señal Enable no se pondrá a '1').